25 | 环境搭建：我们的系统是怎么搭建起来的？

2021-12-14 高楼

《全链路压测实战30讲》 课程介绍 >

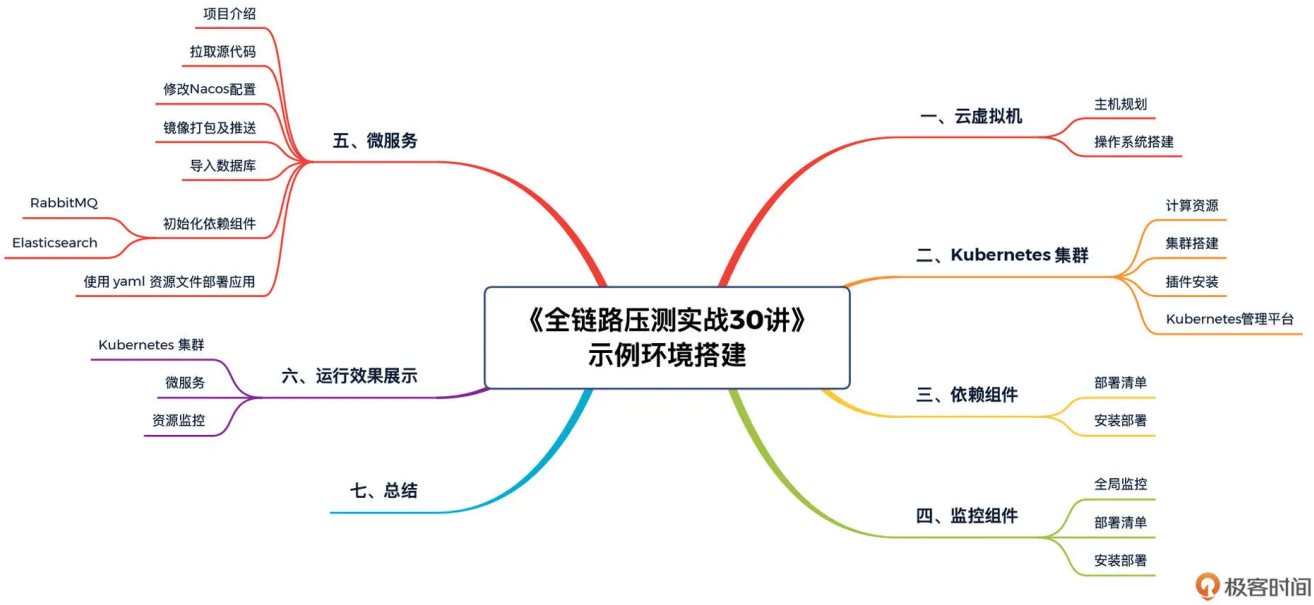
讲述：高楼

时长 04:52 大小 4.46M

你好，我是高楼。

到现在，我们的专栏已经更新了 20 多讲了，在前面的课程中，你应该也已经看到了，为了真正地把全链路压测拉到地面上来，同时也让你更好地理解我的全链路压测理念，我们专门搭建了一个完整的系统，这个专栏所有的内容都是基于这个系统展开的。这节课，我们就来看看，这个系统是怎么搭建起来的。

因为整个系统相对比较复杂，有很多需要考虑、部署的细节，所以这节课的内容会比较长。下面这张图是这节课的目录，你可以先整体了解一下这节课的结构，然后对应这张目录图来学习具体的搭建步骤，以免迷失方向。



我会依照云计算的架构特点，自下向上从云虚拟机到微服务应用，介绍整个搭建的过程。

一. 云虚拟机

云虚拟机负责管理虚拟机的生命周期，包括创建、修改、备份、启停、销毁等。云虚拟机主要是面向资源分配，虚拟机创建出来了云产商就基本没有责任了。至于服务高可用、自动伸缩、监控等这类的功能完全由应用方来处理，云平台不提供支持。云虚拟机适合传统的部署模式，对应用而言和物理机时代没有区别。

1. 主机规划

在这个系统中，我们主要用到了 8 台云虚拟机，下面是具体的硬件配置：

我们可以看到，当前云主机总共使用在应用中的资源是：38 C 的 CPU 资源，88 G 的内存资源。NFS 网络存储服务器不会用在应用中。

关于云虚拟机类型，云产商一般会提供这几个类型：

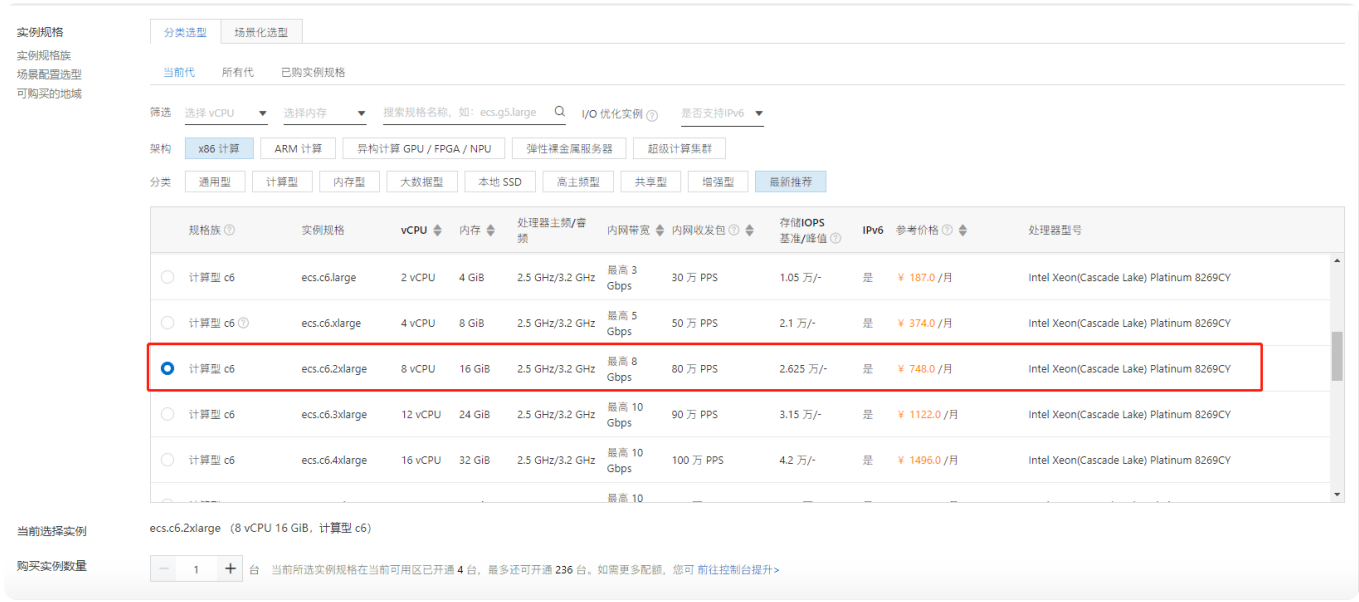
通用均衡型（通常比例 1:4）；

计算密集型（通常比例 1:2）；

内存优化型（通常比例 1:8 以上）；

图形计算型（带有 GPU 能力）。

我们这里选择的云主机类型主要为计算密集型，如下图。



在存储方面，云产商一般会提供高效云盘和 SSD 两种类型，这里我们选择的是 SSD：

你可能会问，这么一套搞下来，成本要多少呢？我算了一下，我们这个项目，所有云主机加在一起一年大概需要六万左右的费用，这其中还包括了云硬盘、云网络等杂七杂八的费用。

我们接着来看下云主机规划。这里我把云主机类型按照节点差异分为两类：

普通节点：

普通节点主要供非被测系统使用，比如压力机、管理平台等。我们可以采用 Docker、二进制等方式来部署。

Kubernetes 节点：

Kubernetes 节点用于部署项目的应用服务，包括 mall-admin、mall-portal、mall-gateway、mall-member、mall-cart 等。另外 Zipkin、Nacos 等基础组件也都采用

Kubernetes 的方式来部署。

具体的节点规划，你可以参考这张表：

主机	操作系统	最低配置	说明
master-1	CentOS 7.8	2C-4G-50G	Kubernetes 控制节点
worker-1	CentOS 7.8	2C-8G-50G	Kubernetes 计算节点
worker-2	CentOS 7.8	2C-8G-50G	Kubernetes计算节点
worker-3	CentOS 7.8	4C-8G-50G	Kubernetes 计算节点
.....	CentOS 7.8	4C-8G-50G	Kubernetes 计算节点（灵活扩展）
nfs-server	CentOS 7.8	2C-4G-500G	网络存储节点
linux-server-1	CentOS 7.8	4C-8G-50G	普通节点（灵活扩展）
.....	CentOS 7.8	4C-8G-50G	普通节点（灵活扩展）

极客时间

在这里，我们规划了一个 Kubernetes 控制节点。如果你计划搭建高可用 Master 集群，至少需要规划三个 Kubernetes 控制节点。至于 Kubernetes 计算节点，结合前面的节点规划，我们在这里配置 7 个 worker 节点，其他的节点会根据需求灵活扩展。

这里我还想提醒一下，如果你所在的项目需要购买云主机的话，最好提早规划。因为我们之前购买的 Kubernetes 集群云主机都是包年包月机型，后来想省钱换成按量付费类型（支持停机后不收费功能）的主机，结果发现网络不互通。这可都是血淋淋的教训。

2. 操作系统搭建

下面，我们再来看下操作系统的搭建。

目前，行业内主流的基于 x86 架构的 Linux 系统，无非是 CentOS 和 Ubuntu 两种。我们选择了 CentOS 系列来搭建 Linux 系统，主要是考虑到了系统的稳定性。CentOS 来自 RedHat 商业版本的重新编译，它在稳定性、系统优化以及兼容性方面，具有比较完善的测试和发版流程。

在 CentOS 7 之后的版本中，CentOS 的内核换成了 Linux 3.x，因此，我们这个课程的分析都是基于 Linux 3.x 这个内核版本展开的。

而大部分的云产商一般会提供下面两种类型的操作系统镜像：

流行成熟的 Linux 发行版

云厂商自有的 Linux 版本（更适配该产商的硬件）

因为我们课程的技术栈不想与云产商绑定，所以我们给每台服务器都安装了流行成熟的 CentOS 7.8 的操作系统。云主机安装操作系统非常简单，创建实例的时候选择对应的系统镜像就可以了：

二. Kubernetes 集群

1. 计算资源

关于集群计算资源，你可以参考这张表：

	裸金属服务器	云虚拟机
I/O 损耗	无	轻微
成本	中	低
额外资源开销	低	中
灵活度	高	低

极客时间

我们在做计算资源规划的时候，通常需要考虑不同的应用场景：

传统虚拟化技术的 I/O 损耗较大，对于 I/O 密集型应用，性能敏感的应用，裸金属服务器相比云虚拟机有更好的性能表现；

在裸金属服务器上部署应用，有更少的额外资源开销（如虚拟化管理、虚拟机操作系统等），并且可以有更高的部署密度，这可以降低基础设施成本；

在裸金属服务器上可以更加灵活地选择网络、存储等设备和软件应用生态。
如果从实际生产环境考虑，我给你的建议是：

对性能极其敏感的应用，如高性能计算，裸金属服务器是较好的选择；

云主机支持热迁移，可以有效降低运维成本；

在工作实践中，可以为 Kubernetes 集群划分静态资源池和弹性资源池，我们就是这样做的。通常而言，静态资源池可以根据需要选择裸金属服务器或者云主机实例；弹性资源池则可以根据应用负载，使用合适规格的云主机实例来优化成本，避免资源浪费，同时提升弹性供给保障。

由于我们这个系统只是课程的示例项目，为了尽可能压榨服务器资源，节省服务器成本，我们选择了云虚拟机的方案。

2. 集群搭建

关于集群搭建，我们的节点规划如下：

复制代码

```
[root@s5 k8s]# kubectl get nodes -o wide
NAME   STATUS   ROLES    AGE    VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IM
s10    Ready    <none>   5d1h   v1.19.5   172.31.184.227   <none>        CentO
s11    Ready    <none>   5d1h   v1.19.5   172.31.184.230   <none>        CentO
s12    Ready    <none>   5d1h   v1.19.5   172.31.184.233   <none>        CentO
s5     Ready    master   5d2h   v1.19.5   172.31.184.225   <none>        CentO
s6     Ready    <none>   5d1h   v1.19.5   172.31.184.226   <none>        CentO
s7     Ready    <none>   5d1h   v1.19.5   172.31.184.224   <none>        CentO
s8     Ready    <none>   5d1h   v1.19.5   172.31.184.229   <none>        CentO
s9     Ready    <none>   5d1h   v1.19.5   172.31.184.228   <none>        CentO
```

集群搭建的具体步骤，你可以参考下面这两个文档进行部署：

单 Master 集群：使用 kubeadm 安装单 master kubernetes 集群（脚本版）

高可用方案： Kubernetes 高可用集群落地二三事

安装的负载均衡组件如下：

组件	版本号	说明
Traefik	2.4	Ingress 能够集成授权入站，并连接到集群服务的规则集合。此外，Ingress 提供七层负载均衡能力，可配置提供外部可访问的 URL、负载均衡、SSL、基于名称的虚拟主机等功能。作为集群流量接入层，Ingress 具有高可靠性。

极客时间

关于 Traefik 搭建的具体步骤，你可以按照下面这个文档进行部署：

Kubernetes 集群部署 Ingress Traefik

如果你没有 Kubernetes 的使用基础，那么我建议学习一下这几篇入门文章：

Kubernetes 集群基本概念

k8s 入门篇 -Kubernetes 的基本概念和术语

k8s 命令篇 -Kubernetes 工作实用命令集结号

Kubernetes 集群常用操作总结

3. 插件安装

我们需要安装的插件主要有三种：网络插件、存储插件和组件。

对于网络插件，我们选用的是目前主流的网络插件 Flannel。如果你的系统有其它选型需求，那你可以参考下面这篇文章，这里我就不做赘述了。

Kubernetes 网络插件（CNI）超过 10Gbit/s 的基准测试结果

如果你在阿里云上安装 k8s，建议使用 Flannel，因为有多个案例表明 Calico 与阿里云存在兼容性问题。

安装 Flannel 插件的具体步骤，你可以参考下文的第 11 章。

使用 kubeadm 安装单 master kubernetes 集群（手动版）

关于存储插件，我们选用的是 NFS 网络存储。因为 NFS 相对简单，上手快，我们只需要部署一个 NFS 服务，再由 Kubernetes 提供一个自动配置卷程序，然后通过 StoageClass 动态配置 Kubernetes 持久就可以了。而且在性能上，NFS 也能满足我们这个系统的需求。

只不过，NFS 并不是高可用方案。如果你是在生产环境中使用，可以考虑把 Ceph 作为存储选型方案。Ceph 是一个统一的分布式存储系统，也是高可用存储方案，并且可以提供比较好的性能、可靠性和可扩展性。但是，Ceph 部署起来更复杂些，同时维护也比 NFS 困难。

我把 NFS 和 Ceph 的详细安装步骤放在这里，你如果有需要，可以参考学习。

NFS： Kubernetes 如何安装 NFS-Subdir-External-Provisioner 存储插件？

Ceph： Kubernetes 集群分布式存储插件 Rook Ceph 部署

另外，不要忘了，NFS 配置中还需要这两个组件：

组件	版本号	说明
nfs-utils	1.3.0	NFS 客户端
NFS-Subdir-External-Provisioner	4.0.0	NFS 自动配置卷程序

极客时间

NFS-Subdir-External-Provisioner 是对 nfs-client-provisioner 的扩展，但是 nfs-client-provisioner 已经不再提供更新，而且 nfs-client-provisioner 的 GitHub 仓库已经迁移到 NFS-Subdir-External-Provisioner 的仓库了。

4. Kubernetes 管理平台

安装组件：

组件	版本号	说明
Kuboard	2.0.5.5	微服务管理工具

极客时间

Kuboard 采用可视化 UI 的方式来管理应用和组件，降低了 Kubernetes 集群的使用门槛。下面我们看看怎么部署 Kuboard 组件。

第一步，k8s 集群执行资源文件：

复制代码

```
kubectl apply -f https://kuboard.cn/install-script/kuboard.yaml
kubectl apply -f https://addons.kuboard.cn/metrics-server/0.3.7/metrics-server
```

复制代码

```
[root@s5 ~]# kubectl apply -f https://kuboard.cn/install-script/kuboard.yaml
deployment.apps/kuboard created
service/kuboard created
serviceaccount/kuboard-user created
clusterrolebinding.rbac.authorization.k8s.io/kuboard-user created
serviceaccount/kuboard-viewer created
clusterrolebinding.rbac.authorization.k8s.io/kuboard-viewer created
[root@s5 ~]#
[root@s5 ~]# kubectl apply -f https://addons.kuboard.cn/metrics-server/0.3.7/m
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delega
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
```

```
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.apps/metrics-server created
service/metrics-server created
[root@s5 ~]#
```

第二步，把 Kuboard 安装好后，我们看一下 Kuboard 的运行状态：

复制代码

```
kubectl get pods -l k8s.kuboard.cn/name=kuboard -n kube-system
```

输出结果：

复制代码

```
[root@s5 ~]# kubectl get pods -l k8s.kuboard.cn/name=kuboard -n kube-system
NAME                        READY   STATUS    RESTARTS   AGE
kuboard-74c645f5df-bsj4k    1/1     Running   0          98s
[root@s5 ~]#
```

这个结果表明 kuboard 已经部署成功了。

接着，我们获取管理员 Token 。这一步是为了登录访问 Kuboard，检查组件是否成功运行。

复制代码

```
# 可在第一个 Master 节点上执行此命令
echo $(kubectl -n kube-system get secret $(kubectl -n kube-system get secret |
```

复制代码

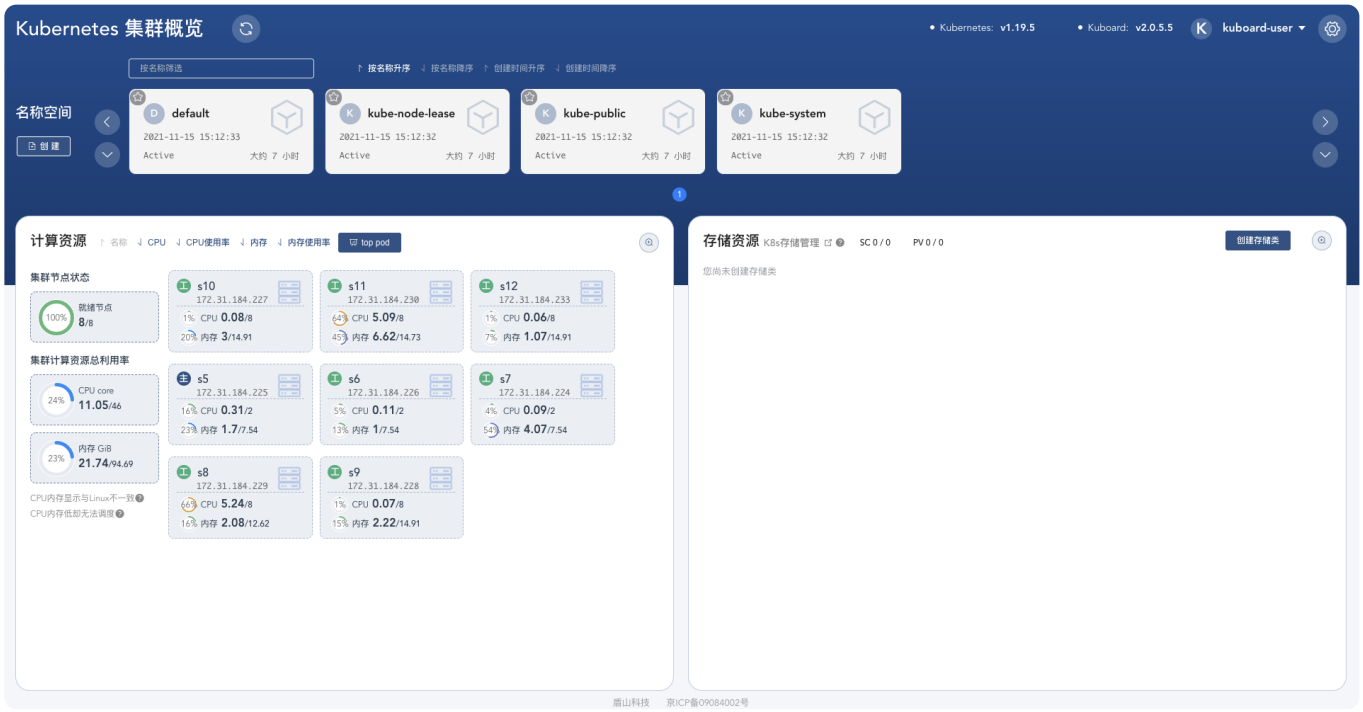
```
[root@s5 k8s]# echo $(kubectl -n kube-system get secret $(kubectl -n kube-syst
eyJhbGciOiJSUzI1NiIsImtpZCI6IldTYlJYVXIzbWZNUnBSczF1ZFc5cmRQbEp2UlFtbXREZDVCSE
```

通过检查部署我们了解到，Kuboard Service 使用了 NodePort 的方式暴露服务，NodePort 为 32567。因此，我们可以按照下面这个方式访问 Kuboard：

```
1 http://任意一个Worker节点的IP地址:32567/
```

复制代码

然后，在登录界面中输入管理员 Token，这样就可以进入到 Kuboard 集群的概览页了。



注意，如果你使用的是阿里云、腾讯云等云服务，那么你可以在对应的安全组设置里，开放 worker 节点 32567 端口的入站访问，你也可以修改 Kuboard.yaml 文件，使用自己定义的 NodePort 端口号。

三. 依赖组件

1. 部署清单

组件	版本号	说明
MySQL	5.7	关系型数据库
Redis	5.0	分布式缓存
Elasticsearch	7.6.2	搜索引擎
Kibana	7.6.2	日志可视化看板
MongoDB	4.2.5	文档数据库
RabbitMQ	3.7.14	消息队列
Helm 3	3.2.4	Kubernetes 包管理器
Nacos	1.4.0	服务注册及配置中心
Harbor	2.0.5.5	镜像仓库
Sentinel	1.8.0	系统保护

极客时间

2. 安装部署

对于上述依赖组件的安装部署，我整理了对应的教程放在这里，如果有兴趣你可以尝试一下。

MySQL 的二进制安装方式，在网上的教程多如牛毛，我在这里就不介绍了，如果你想知道怎么在 Kubernetes 下部署 MySQL，你可以参考这个链接中的详细步骤：

- Kubernetes 集群部署 MySQL+ mysqld_exporter (单节点)
- 如何在 Kubernetes 集群中搭建一个复杂的 MySQL 数据库

Elasticsearch 集群的部署可以参考：

- Kubernetes Helm3 部署 Elasticsearch & Kibana 7 集群

Weave Scope 的部署可以参考：

- Kubernetes 集群可视化监控之 Weave Scope 入门

镜像仓库 Harbor 的部署可以参考：

Kubernetes 集群仓库 harbor Helm3 部署

Nacos 的部署可以参考：

Docker 单机模式： Nacos Docker 快速开始

Kubernetes：Kubernetes 集群部署 nacos 1.4（集群版）

Redis、RabbitMQ、MongoDB 单机部署的部署可以参考：

Kubernetes 集群部署 Redis + redis_exporter (单点)

Kubernetes 集群部署 RabbitMQ + exporter (单节点)

Kubernetes 集群部署 MongoDB+exporter (单节点)

四. 监控组件

1. 全局监控

不知道你还记不记得我们这个系统的架构：

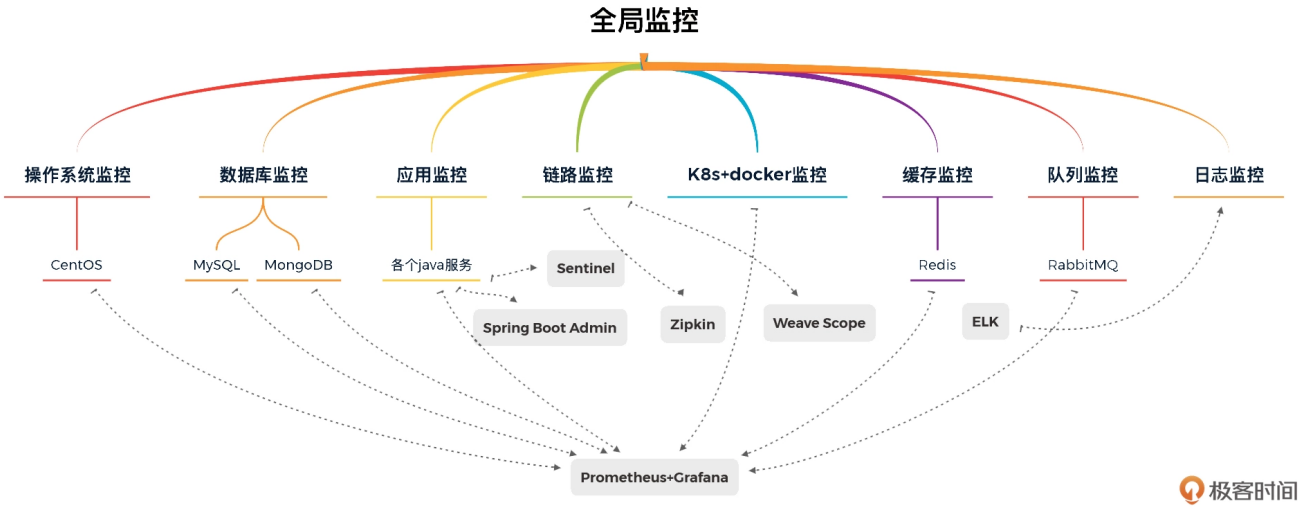
根据这个系统的架构，我们选择的工具需要监控到这几个层面：

第一层，云虚拟机；

第二层，Kubernetes 套件；

第三层，各种应用所需要的技术组件。

其实，有了上面的系统架构，监控设计大体的轮廓就有了。对于我们这个课程所用的系统，全局监控如下所示：



从上图来看，我们使用 Prometheus/Grafana/Spring Boot Admin/Zipkin/Weave Scope/ELFK/Sentinel 就可以实现具有全局视角的第一层监控。对于工具中没有覆盖的第一层计数器，我们只能在执行场景时再执行命令来补充了。

2. 部署清单

组件	版本号	说明
Zpkin	8.1.0	分布式跟踪系统
Prometheus	0.6.0	资源监控系统
Weave Scope	1.13.2	Kubernetes 可视化监控工具
redis_exporter	1.23.0	Redis 资源监控
mysqld_exporter	0.12.1	MySQL 数据库资源监控
mongodb_exporter	0.20.4	MongoDB 资源监控
rabbitmq_exporter	1.0.0	消息队列资源监控

极客时间

3. 安装部署

对于上面这些监控工具的部署，我也把相应的安装教程放在这里供你参考学习。

Kubernetes 集群资源监控的部署：

- Kubernetes 集群监控 kube-prometheus 部署
- Kubernetes 集群监控 controller-manager & scheduler 组件
- Kubernetes 集群监控 ETCD 组件

日志聚合部署的部署：

- Kubernetes 集群日志监控 EFK 安装

依赖组件的部署：

- Kubernetes 集群监控 kube-prometheus 自动发现

APM 链路跟踪的部署：

Kubnernetes 集群部署 Zipkin+Kafka+ElasticSearch 实现链路追踪

五. 微服务

1. 项目介绍

在搭建这个课程所用的系统时，我采用了微服务的架构，这也是当前主流的技术架构。

如果你有兴趣了解详细的项目介绍，可以参考这篇文章：《微服务电商项目技术全解析》。这里面主要介绍了该项目的一些预备知识、系统结构、主要技术栈以及核心组件。此外，还有相关的运行效果截图。

2. 拉取源代码

我们把 Git clone 项目源代码下载到本地，来部署我们的被测系统：

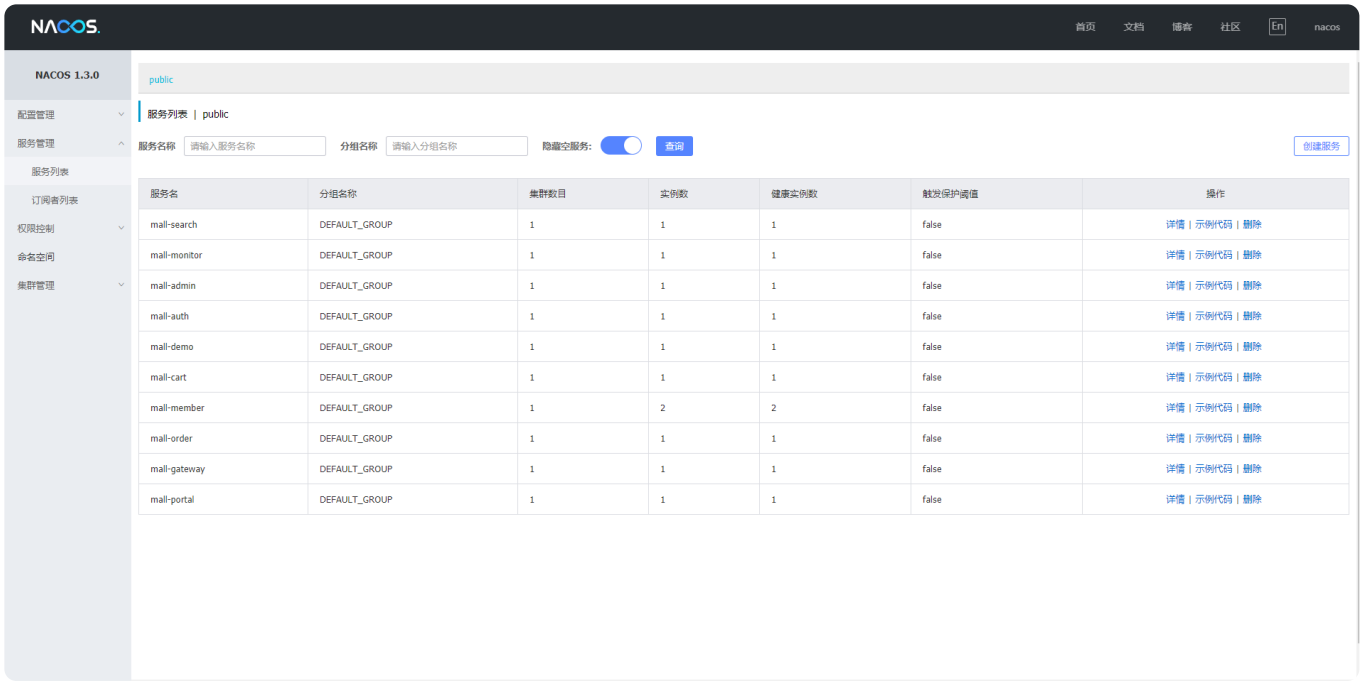
```
1 git clone https://github.com/xncssj/7d-mall-microservice.git
```

复制代码

3. 修改 Nacos 配置

我们先将项目 config 目录下的配置包导入到 Nacos 中，然后根据自己的实际需要修改相关配置。

接着，我们将配置信息导入到 Nacos 中后，会显示这样的信息：



请注意，我们修改的配置文件主要是每个单体服务下的 application-prod.yml 和 bootstrap-prod.yml。因为这两个全局配置文件，都是服务容器内加载的配置文件。

4. 镜像打包及推送

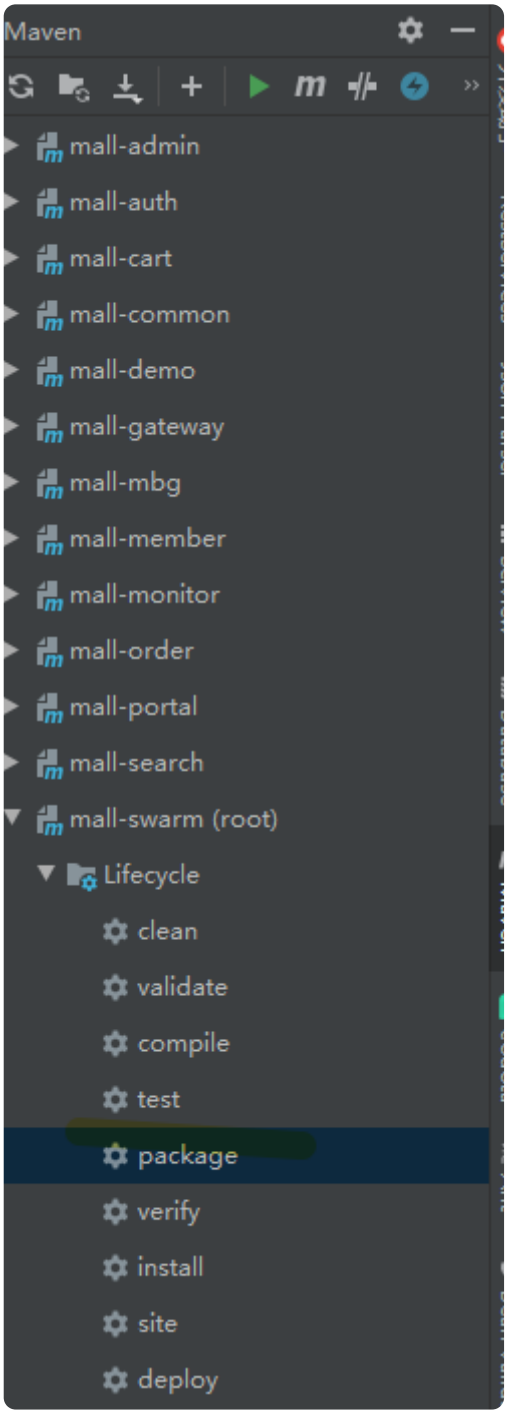
我们使用 Java 语言的 IDE （推荐 IDEA ）打开项目工程。

首先，修改项目根目录下的 pom.xml 文件：

复制代码

```
<properties>
    <!--改为你自己的 Docker 服务远程访问地址-->
    <docker.host>http://172.16.106.237:2375</docker.host>
</properties>
```

在 IDEA 的右边 Maven 标签页，我们可以找到 root 工程下的 package 按钮，选中并执行：



然后，在编译的远程 Docker 主机上，我们修改所有服务的镜像标签名称。之后，再推送镜像到 Docker 仓库。

5. 导入数据库

这一步需要将项目 document/sql 目录下的 SQL 脚本导入到 MySQL 数据库中。

6. 初始化依赖组件

6.1 RabbitMQ

第一步，进入 RabbitMQ 容器并开启管理功能：

复制代码

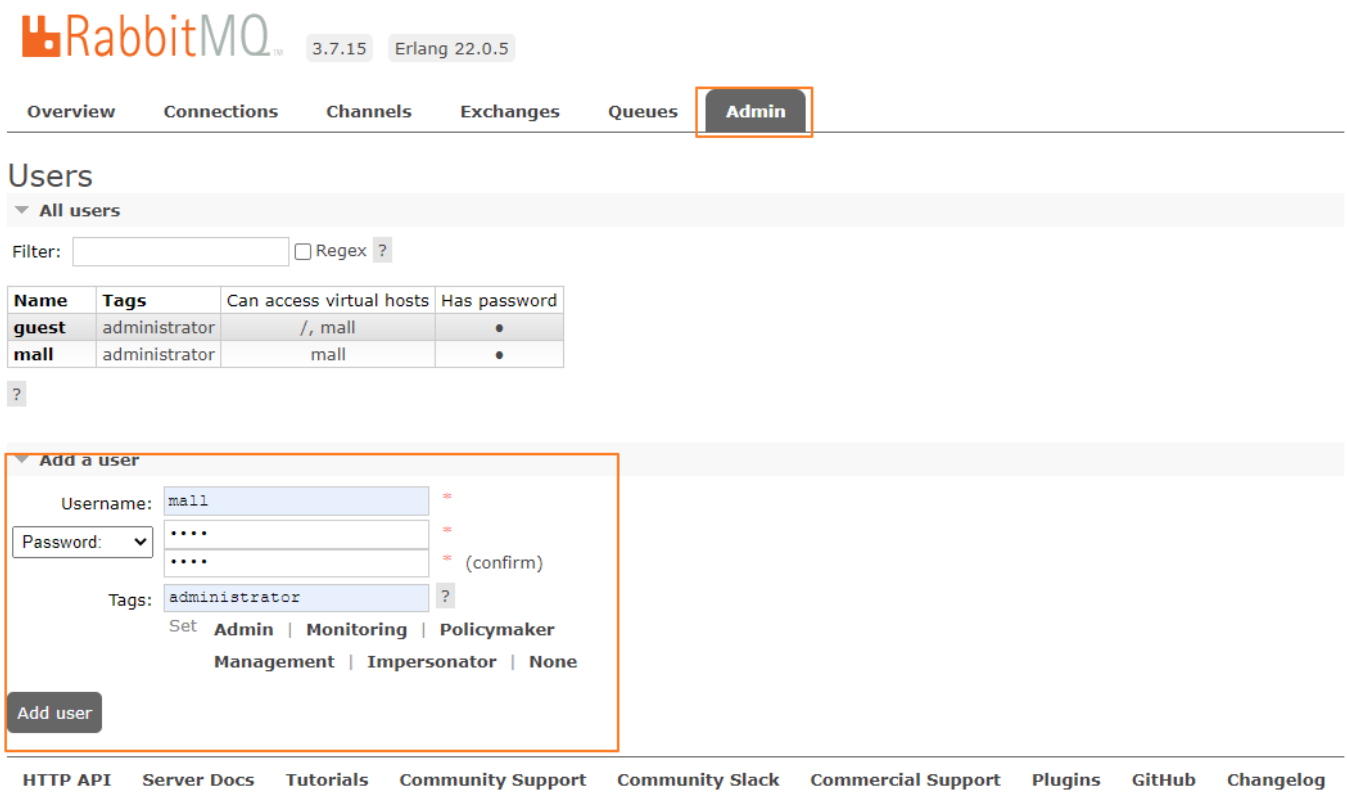
```
#登录容器的时候需要注意到容器支持的 shell 是什么。
kubectl exec -it <pod-name> -n <ns-name> bash
kubectl exec -it <pod-name> -n <ns-name> sh

root@cloud-rabbitmq-5b49d784c-gbr8m:/# rabbitmq-plugins enable rabbitmq_manage
Enabling plugins on node rabbit@cloud-rabbitmq-5b49d784c-gbr8m:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@cloud-rabbitmq-5b49d784c-gbr8m...
Plugin configuration unchanged.
```

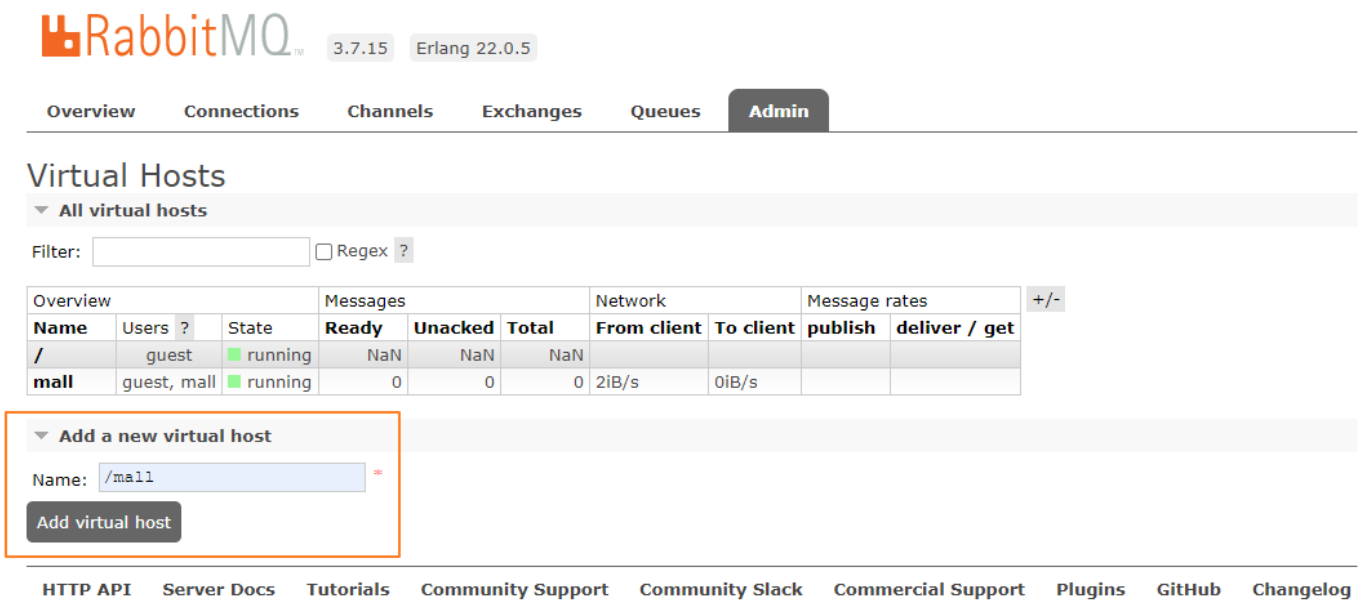
因为 RabbitMQ Service 使用 NodePort 的方式暴露控制台地址，比如 NodePort 为 15672。所以，第二步，我们访问地址 http:// 计算节点 IP:15672/ 地址，查看是否安装成功：

第三步，输入账号密码并登录 guest/guest。

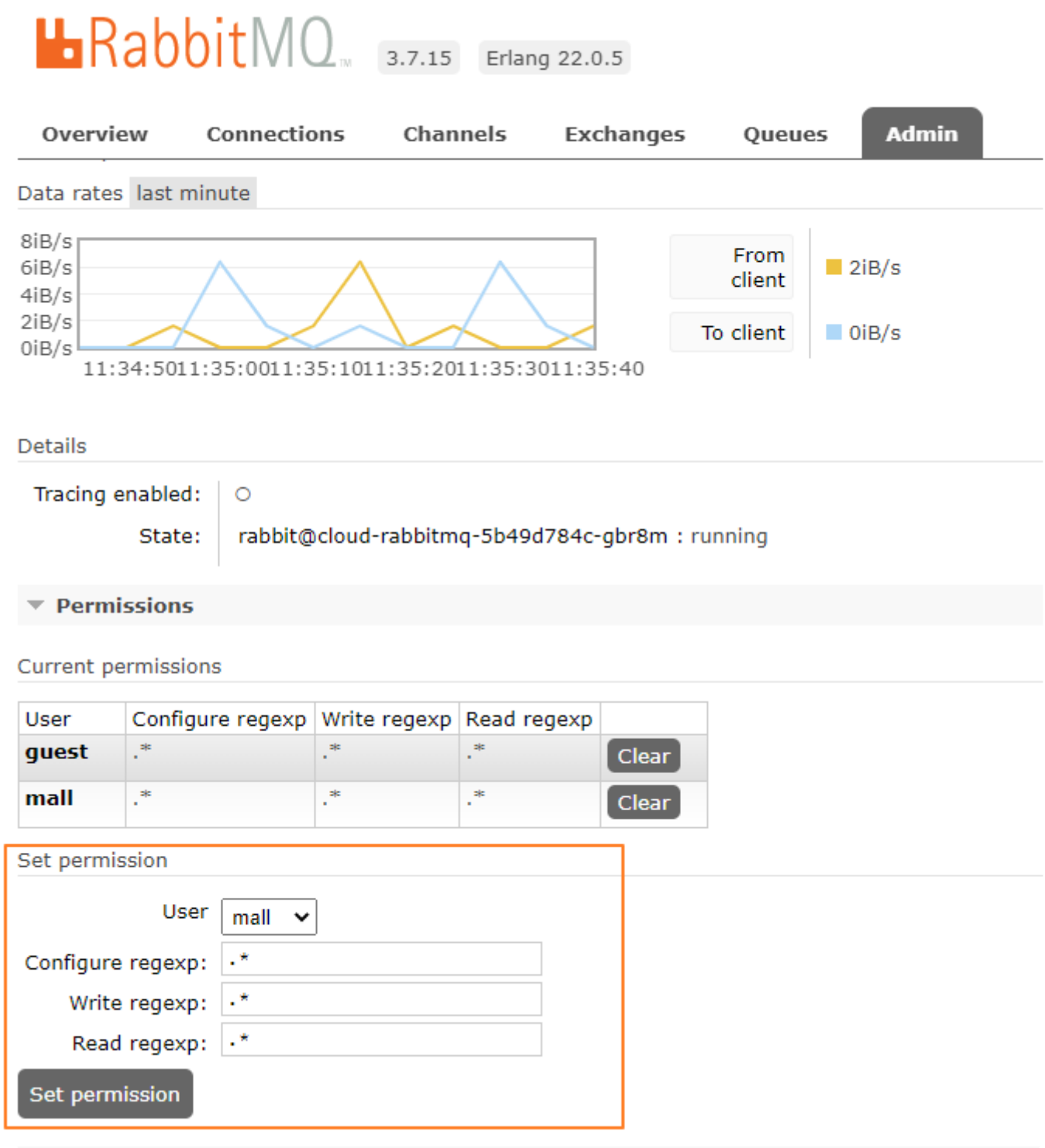
第四步，创建帐号并设置其角色为管理员 mall/mall。



第五步，创建一个新的虚拟 host 为 /mall。



第六步，点击 mall 用户进入用户配置页面，给 mall 用户配置该虚拟 host 的权限。



到这里，RabbitMQ 的初始化就完成了。

6.2 Elasticsearch

安装中文分词器 IKAnalyzer，并重新启动：

```
#此命令需要在容器中运行
elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-i
```

7. 使用 yaml 资源文件部署应用

将项目 document/k8s 目录下的 yaml 资源文件中的 Dokcer 镜像，修改为自己的 Tag 并上传到 k8s 集群中执行：

```
kubectl apply -f k8s/
```

六. 运行效果展示

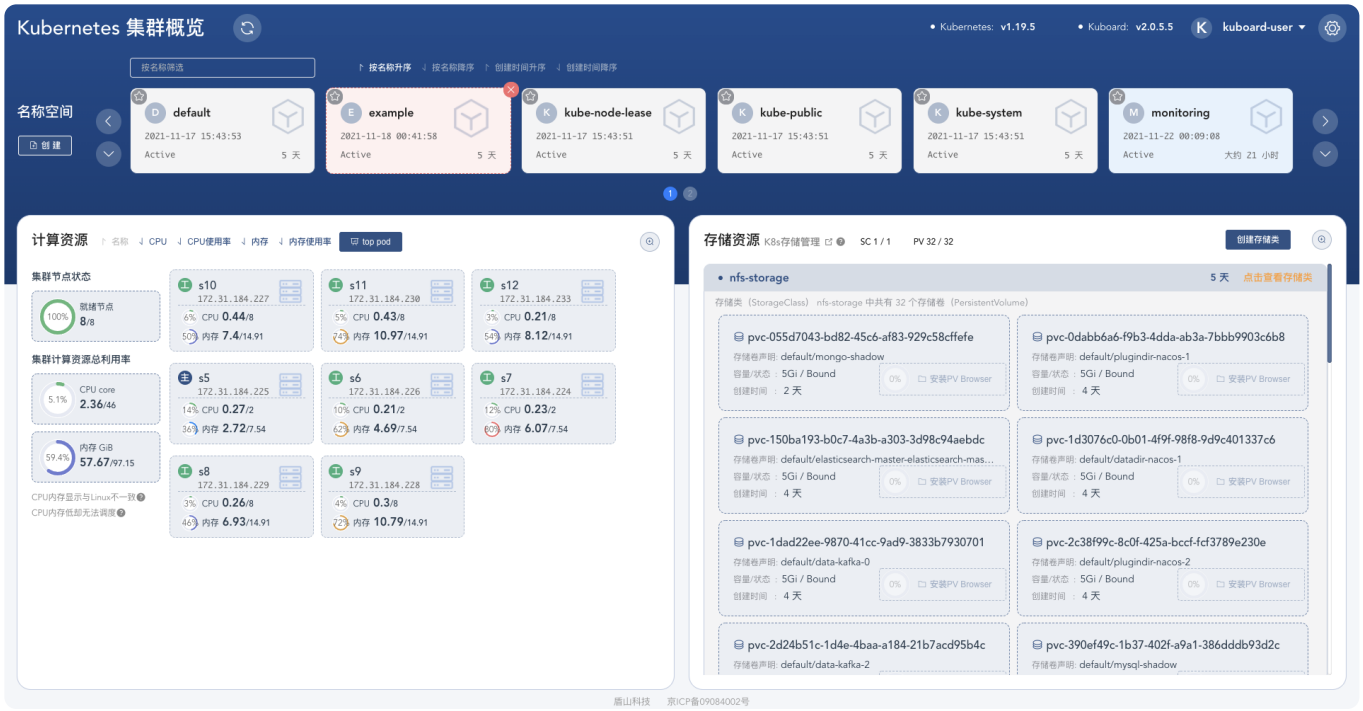
前面的环境搭建完成了，最后我们一起来看下运行效果。

1. Kubernetes 集群

Kubernetes 集群：

```
~ ▶ kubectl get nodes
NAME   STATUS   ROLES    AGE    VERSION
s10    Ready    <none>   5d8h   v1.19.5
s11    Ready    <none>   5d8h   v1.19.5
s12    Ready    <none>   5d8h   v1.19.5
s5     Ready    master   5d8h   v1.19.5
s6     Ready    <none>   5d8h   v1.19.5
s7     Ready    <none>   5d8h   v1.19.5
s8     Ready    <none>   5d8h   v1.19.5
s9     Ready    <none>   5d8h   v1.19.5
~ ▶
```

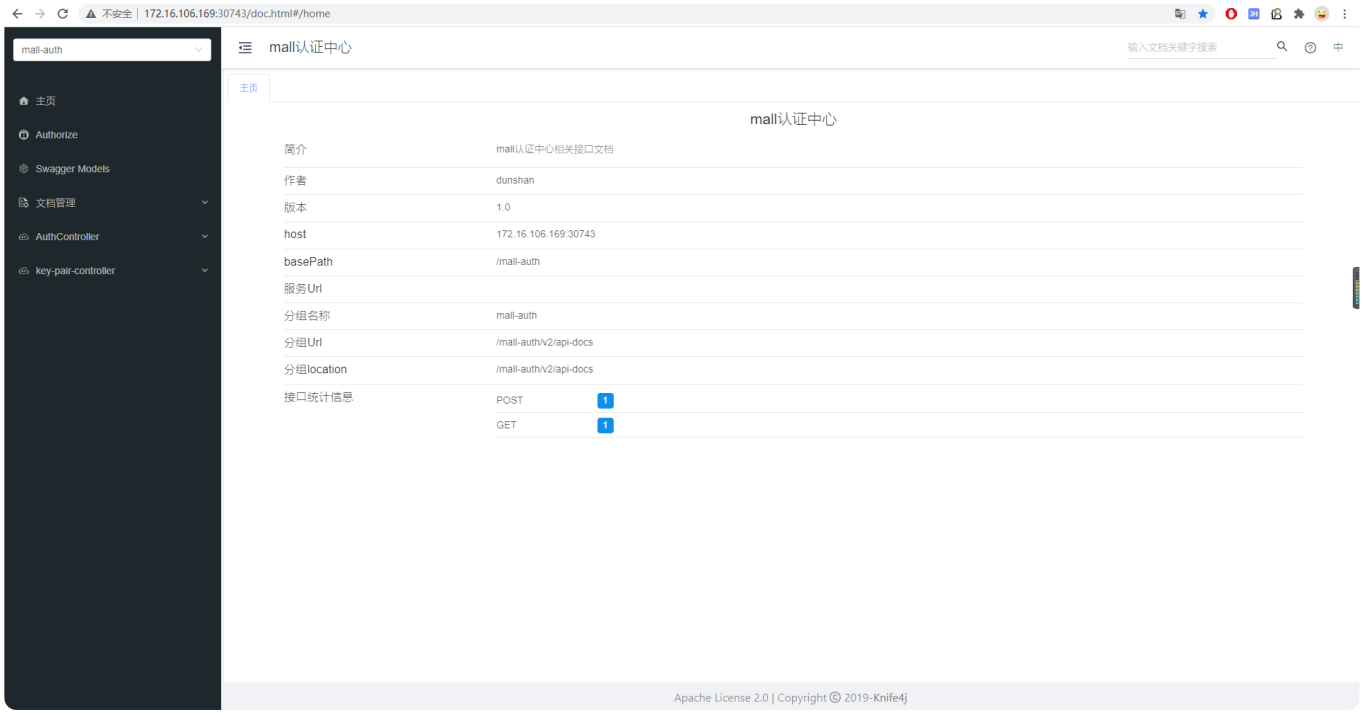
微服务管理：



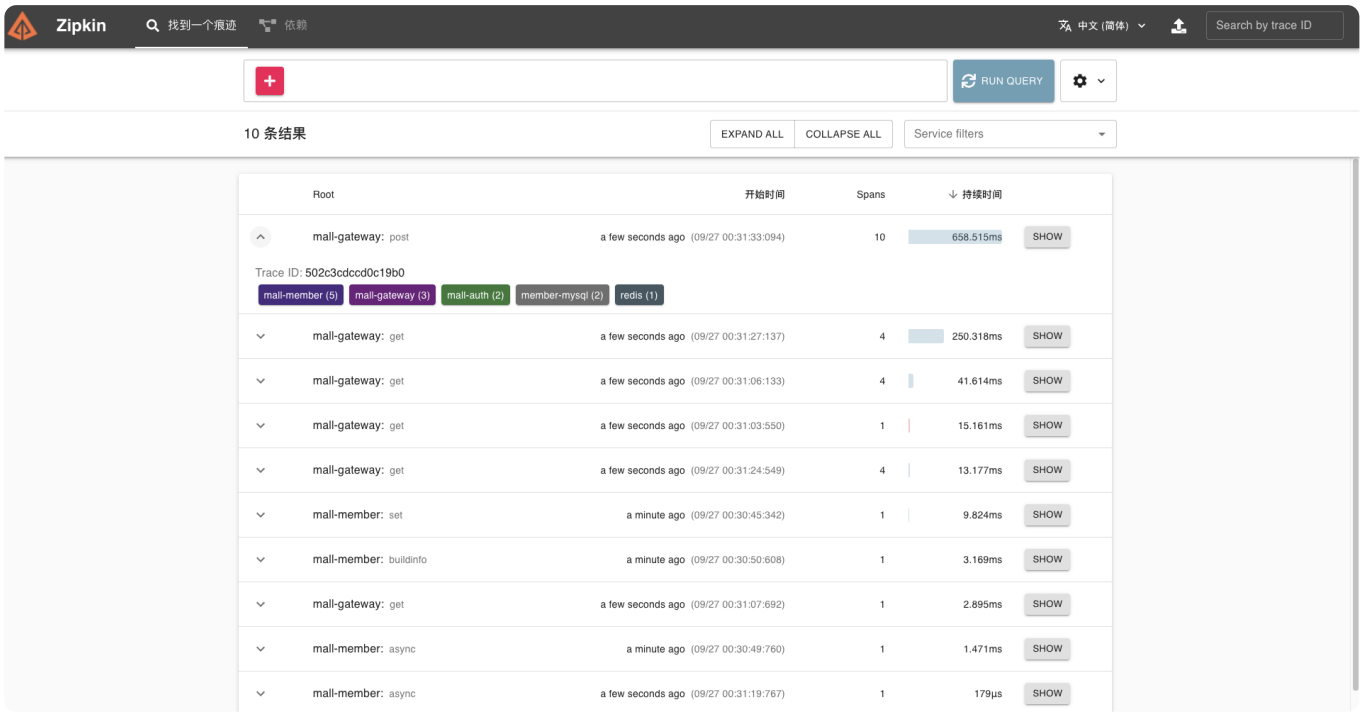
2. 微服务

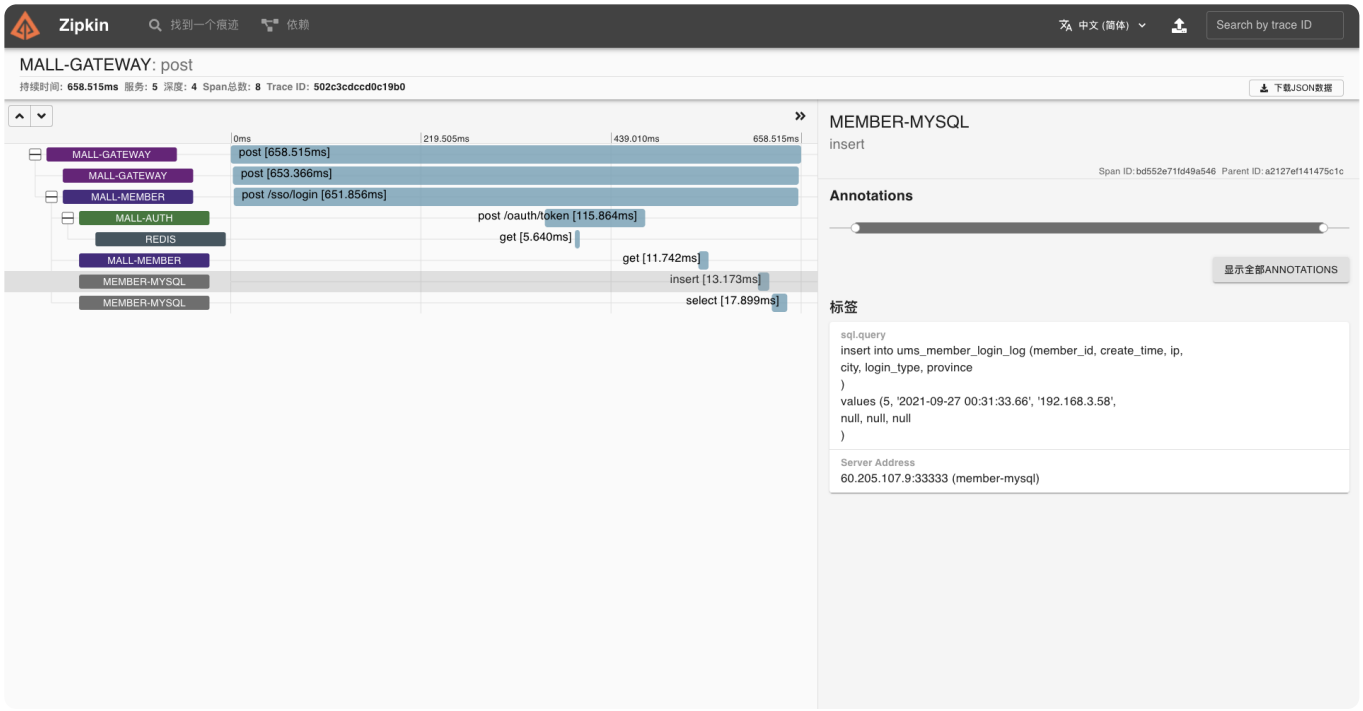
部署架构图：

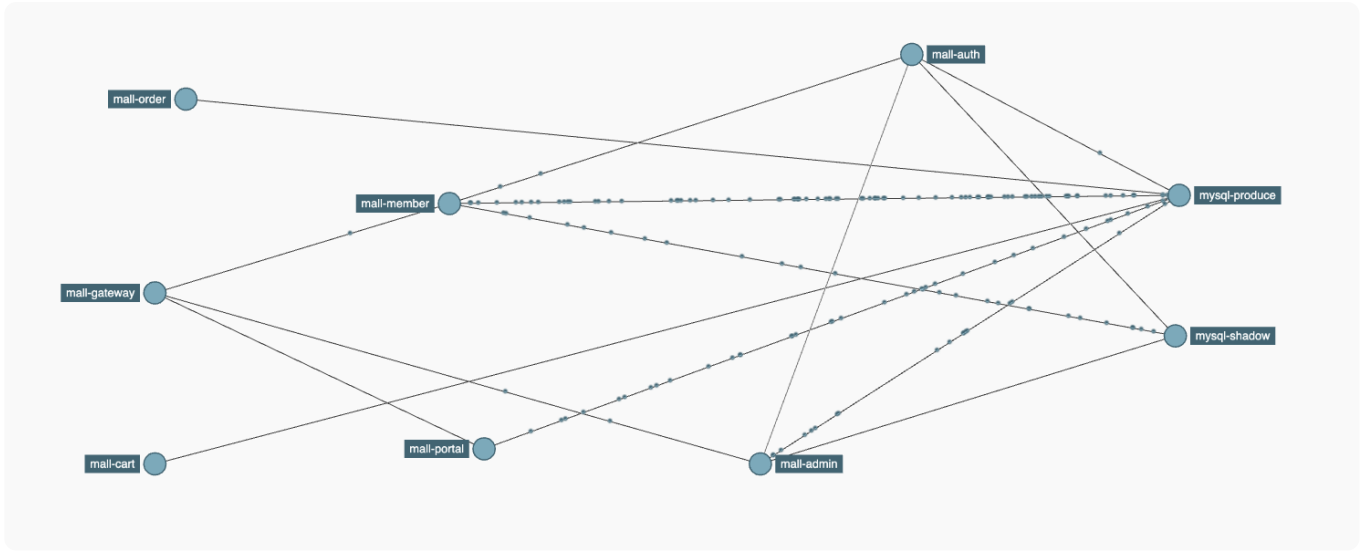
API 文档：


不安全 | 172.16.106.169:30743/doc.html#/home
mall-auth
mall认证中心
主页
Authorize
Swagger Models
文档管理
AuthController
key-pair-controller
mall认证中心
简介 mall认证中心相关接口文档
作者 dunshan
版本 1.0
host 172.16.106.169:30743
basePath /mall-auth
服务Url
分组名称 mall-auth
分组Url /mall-auth/v2/api-docs
分组location /mall-auth/v2/api-docs
接口统计信息
POST 1
GET 1
Apache License 2.0 | Copyright © 2019-Knife4j

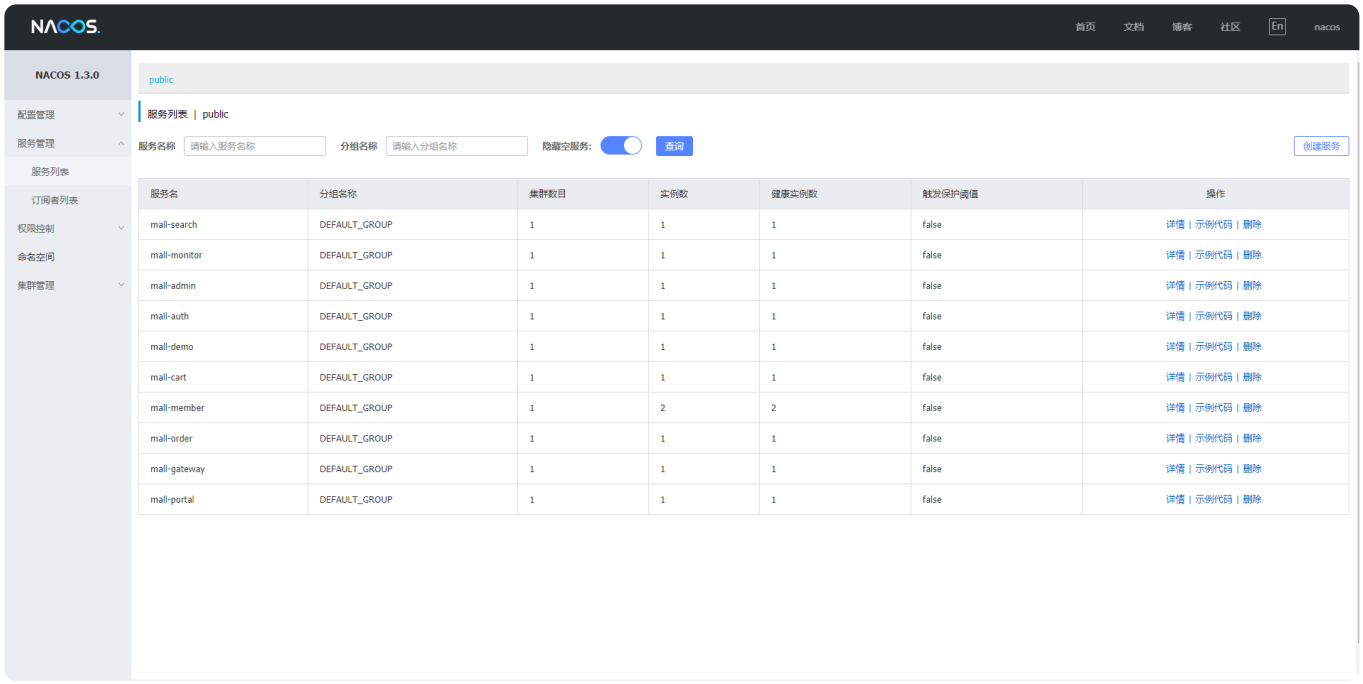
调用链监控：


Zipkin
找到一个痕迹
依赖
中文 (简体)
Search by trace ID
RUN QUERY
10 条结果
EXPAND ALL
COLLAPSE ALL
Service filters
Root
开始时间
Spans
持续时间
mall-gateway: post a few seconds ago (09/27 00:31:33:094) 10 658.515ms SHOW
Trace ID: 502c3cdccd0c19b0
mall-member (5)
mall-gateway (3)
mall-auth (2)
member-mysql (2)
redis (1)
mall-gateway: get a few seconds ago (09/27 00:31:27:137) 4 250.318ms SHOW
mall-gateway: get a few seconds ago (09/27 00:31:06:133) 4 41.614ms SHOW
mall-gateway: get a few seconds ago (09/27 00:31:03:550) 1 15.161ms SHOW
mall-gateway: get a few seconds ago (09/27 00:31:24:549) 4 13.177ms SHOW
mall-member: set a minute ago (09/27 00:30:45:342) 1 9.824ms SHOW
mall-member: buildinfo a minute ago (09/27 00:30:50:608) 1 3.169ms SHOW
mall-gateway: get a few seconds ago (09/27 00:31:07:692) 1 2.895ms SHOW
mall-member: async a minute ago (09/27 00:30:49:760) 1 1.471ms SHOW
mall-member: async a few seconds ago (09/27 00:31:19:767) 1 179µs SHOW

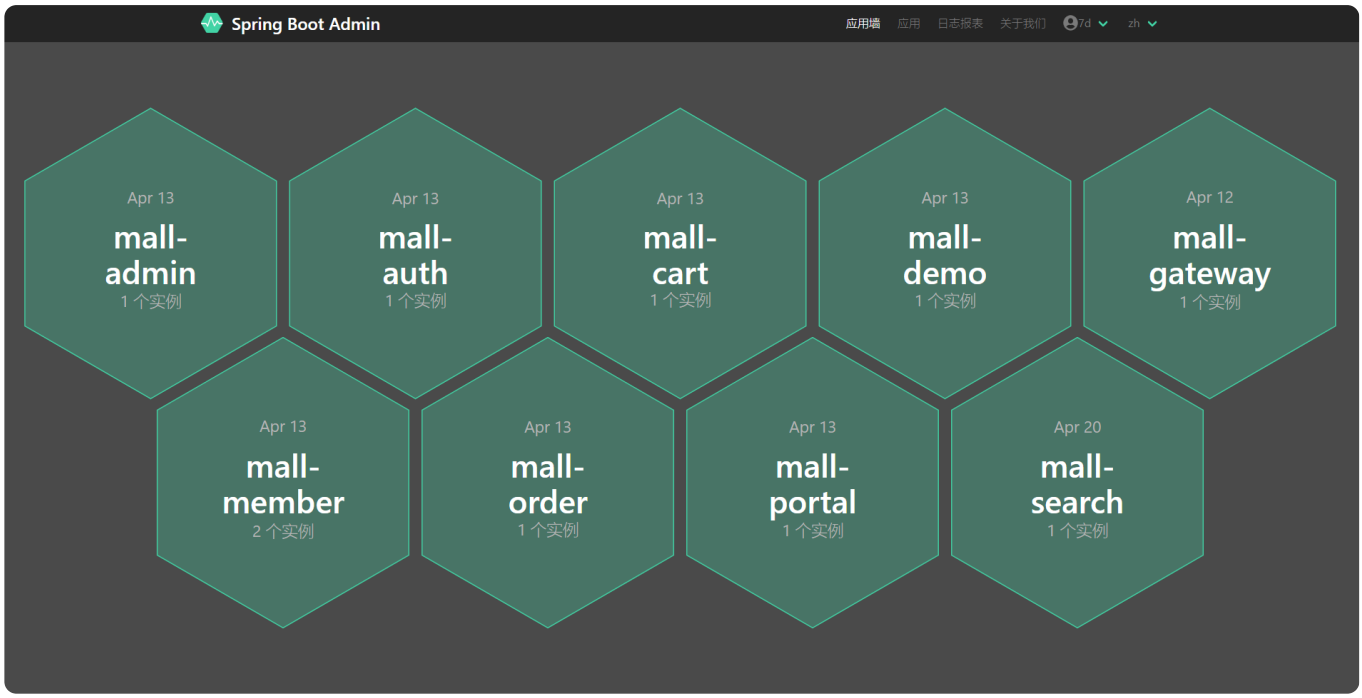

Zipkin
找到一个痕迹
依赖
中文 (简体)
Search by trace ID
MALL-GATEWAY: post
持续时间: 658.515ms 服务: 5 深度: 4 Span总数: 8 Trace ID: 502c3cdccd0c19b0
下载JSON数据
0ms
219.505ms
439.010ms
658.515ms
MALL-GATEWAY
post [658.515ms]
MALL-GATEWAY
post [653.366ms]
MALL-MEMBER
post /sso/login [651.856ms]
MALL-AUTH
post /oauth/token [115.864ms]
REDIS
get [5.640ms]
MALL-MEMBER
get [11.742ms]
MEMBER-MYSQL
insert [13.173ms]
MEMBER-MYSQL
select [17.899ms]
MEMBER-MYSQL
insert
Span ID: bd552e71fd49a546 Parent ID: a2127ef141475c1c
Annotations
显示全部ANNOTATIONS
标签
sql.query
insert into ums_member_login_log (member_id, create_time, ip,
city, login_type, province
)
values (5, '2021-09-27 00:31:33.66', '192.168.3.58',
null, null, null
)
Server Address
60.205.107.9:33333 (member-mysql)


mall-auth
mall-order
mall-member
mysql-produce
mall-gateway
mysql-shadow
mall-cart
mall-portal
mall-admin

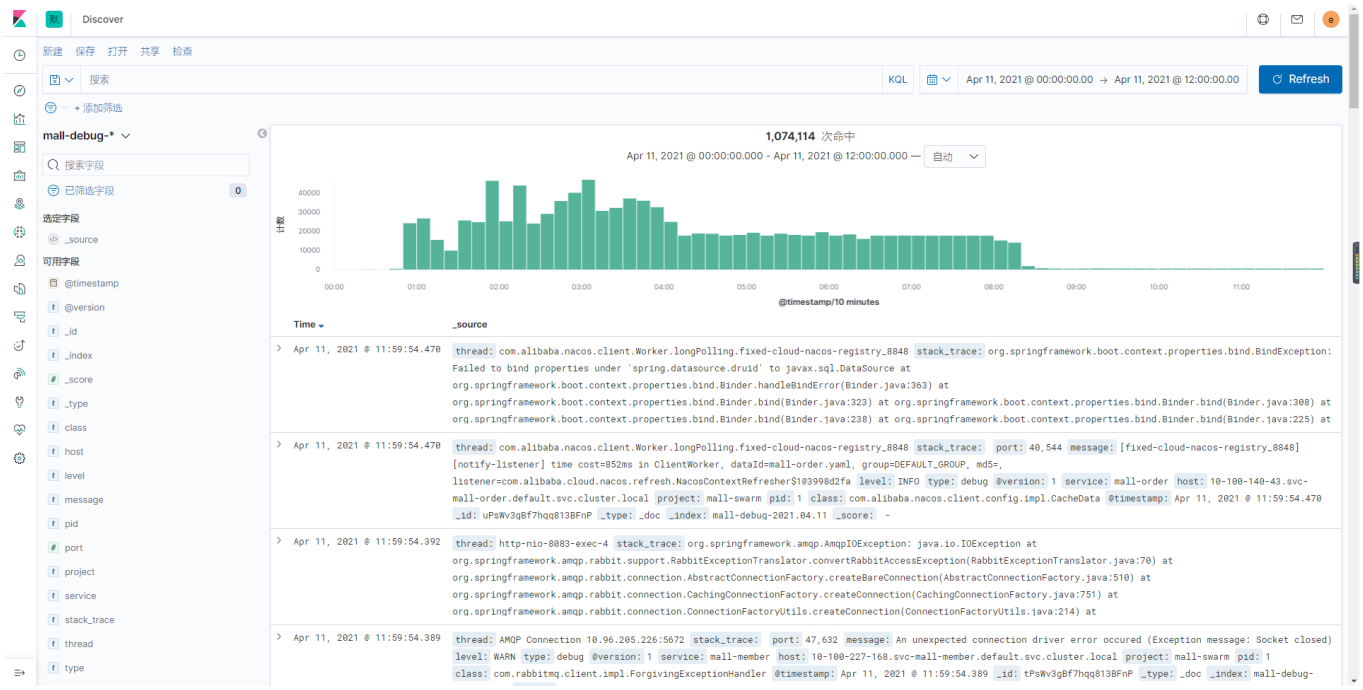
服务注册：


NACOS
首页 文档 博客 社区 En nacos
NACOS 1.3.0
配置管理
服务管理
服务列表
订阅者列表
权限控制
命名空间
集群管理
public
服务列表 | public
服务名称 请输入服务名称
分组名称 请输入分组名称
隐藏空服务:
查询
创建服务
服务名 分组名称 集群数目 实例数 健康实例数 触发保护阈值 操作
mall-search DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-monitor DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-admin DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-auth DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-demo DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-cart DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-member DEFAULT_GROUP 1 2 2 false 详情 | 示例代码 | 删除
mall-order DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-gateway DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-portal DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除

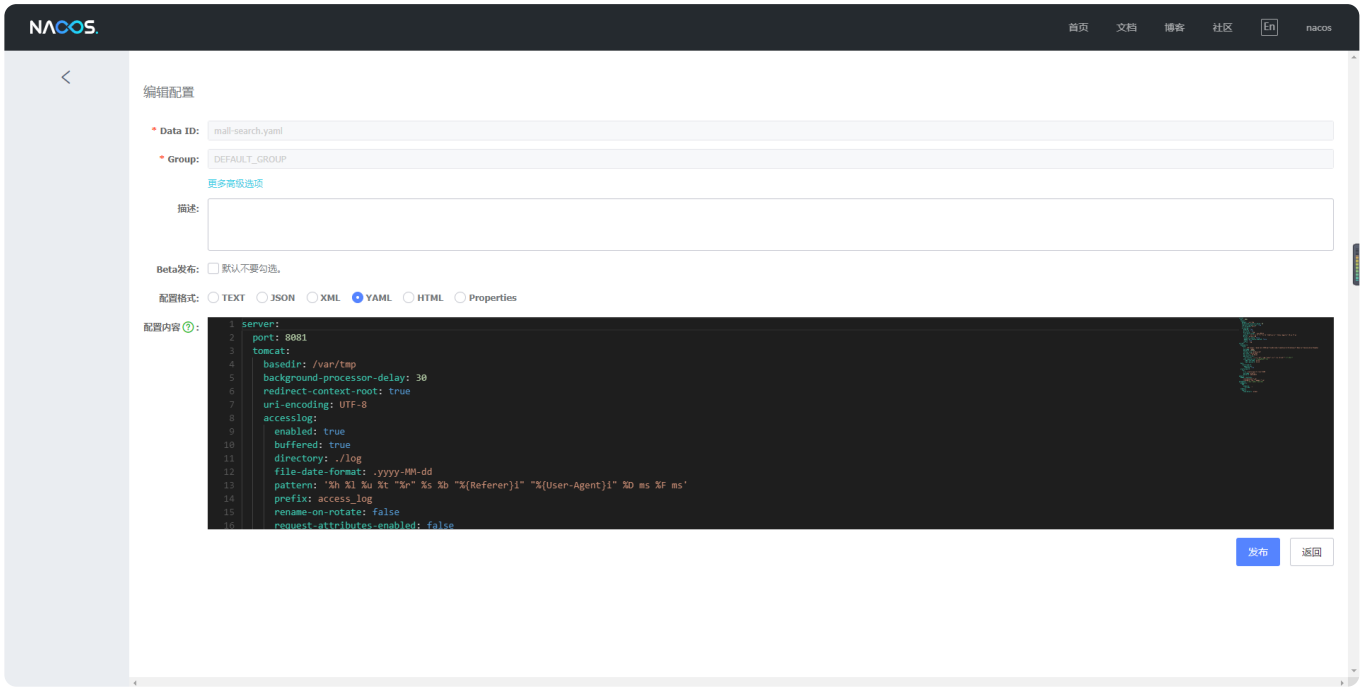
服务监控:


Spring Boot Admin
应用墙 应用 日志报表 关于我们 7d zh
Apr 13
mall-admin
1 个实例
Apr 13
mall-auth
1 个实例
Apr 13
mall-cart
1 个实例
Apr 13
mall-demo
1 个实例
Apr 12
mall-gateway
1 个实例
Apr 13
mall-member
2 个实例
Apr 13
mall-order
1 个实例
Apr 13
mall-portal
1 个实例
Apr 20
mall-search
1 个实例

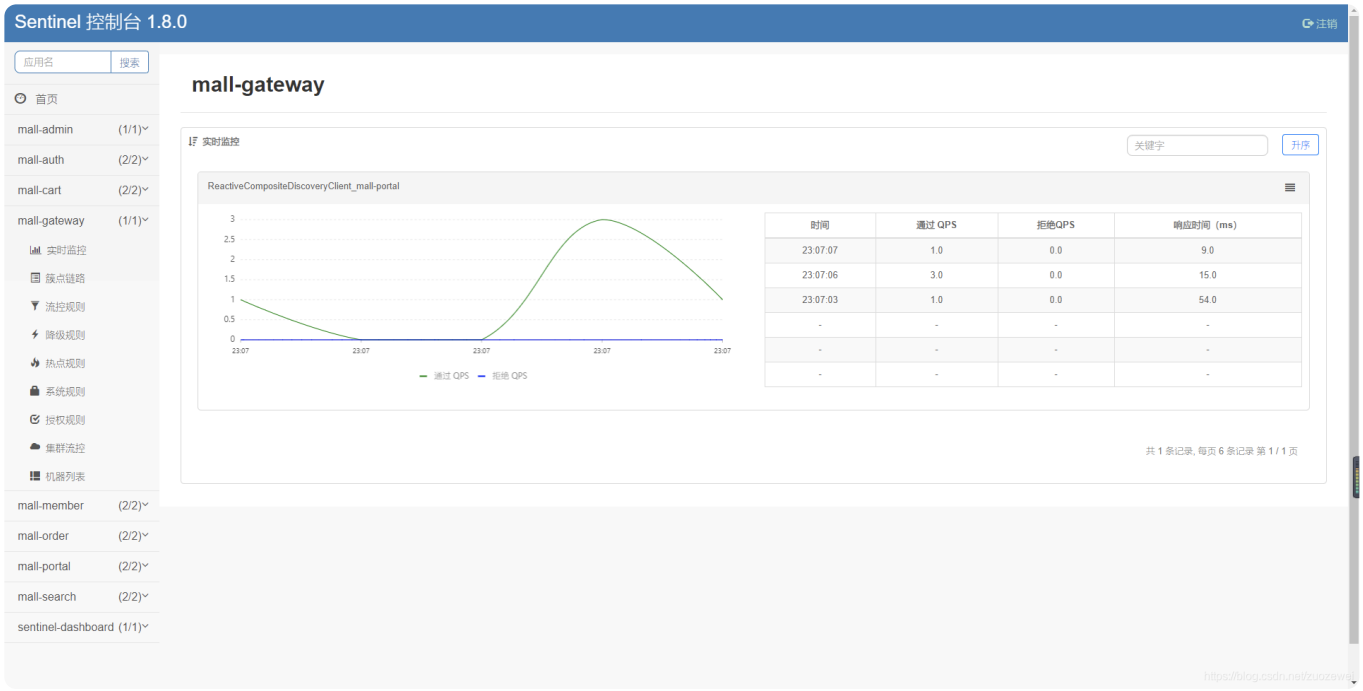
日志聚合：


Discover
新建 保存 打开 共享 检查
搜索
KQL
Apr 11, 2021 @ 00:00:00.00 → Apr 11, 2021 @ 12:00:00.00
Refresh
+ 添加筛选
mall-debug-*
搜索字段
已筛选字段 0
选定字段
_source
可用字段
@timestamp
@version
_id
_index
_score
_type
class
host
level
message
pid
port
project
service
stack_trace
thread
type
1,074,114 次命中
Apr 11, 2021 @ 00:00:00.000 - Apr 11, 2021 @ 12:00:00.000 — 自动
计数
@timestamp/10 minutes
Time
_source
Apr 11, 2021 @ 11:59:54.470
thread: com.alibaba.nacos.client.Worker.longPolling.fixed-cloud-nacos-registry_8848 stack_trace: org.springframework.boot.context.properties.bind.BindException: Failed to bind properties under 'spring.datasource.druid' to javax.sql.DataSource at org.springframework.boot.context.properties.bind.Binder.handleBindError(Binder.java:363) at org.springframework.boot.context.properties.bind.Binder.bind(Binder.java:323) at org.springframework.boot.context.properties.bind.Binder.bind(Binder.java:308) at org.springframework.boot.context.properties.bind.Binder.bind(Binder.java:238) at org.springframework.boot.context.properties.bind.Binder.bind(Binder.java:225) at
Apr 11, 2021 @ 11:59:54.470
thread: com.alibaba.nacos.client.Worker.longPolling.fixed-cloud-nacos-registry_8848 stack_trace: port: 40,544 message: [fixed-cloud-nacos-registry_8848] [notify-listener] time cost=852ms in ClientWorker, dataId=mall-order.yaml, group=DEFAULT_GROUP, md5=, listener=com.alibaba.cloud.nacos.refresh.NacosContextRefresher$103998d2fa level: INFO type: debug @version: 1 service: mall-order host: 10-100-140-43.svc-mall-order.default.svc.cluster.local project: mall-swarm pid: 1 class: com.alibaba.nacos.client.config.impl.CacheData @timestamp: Apr 11, 2021 @ 11:59:54.470 _id: uPsWv3gBf7hqq813BFnP _type: _doc _index: mall-debug-2021.04.11 _score: -
Apr 11, 2021 @ 11:59:54.392
thread: http-nio-8083-exec-4 stack_trace: org.springframework.amqp.AmqpIOException: java.io.IOException at org.springframework.amqp.rabbit.support.RabbitExceptionTranslator.convertRabbitAccessException(RabbitExceptionTranslator.java:70) at org.springframework.amqp.rabbit.connection.AbstractConnectionFactory.createBareConnection(AbstractConnectionFactory.java:510) at org.springframework.amqp.rabbit.connection.CachingConnectionFactory.createConnection(CachingConnectionFactory.java:751) at org.springframework.amqp.rabbit.connection.ConnectionFactoryUtils.createConnection(ConnectionFactoryUtils.java:214) at
Apr 11, 2021 @ 11:59:54.389
thread: AMQP Connection 10.96.205.226:5672 stack_trace: port: 47,632 message: An unexpected connection driver error occured (Exception message: Socket closed) level: WARN type: debug @version: 1 service: mall-member host: 10-100-227-168.svc-mall-member.default.svc.cluster.local project: mall-swarm pid: 1 class: com.rabbitmq.client.impl.ForgivingExceptionHandler @timestamp: Apr 11, 2021 @ 11:59:54.389 _id: tPsWv3gBf7hqq813BFnP _type: _doc _index: mall-debug-

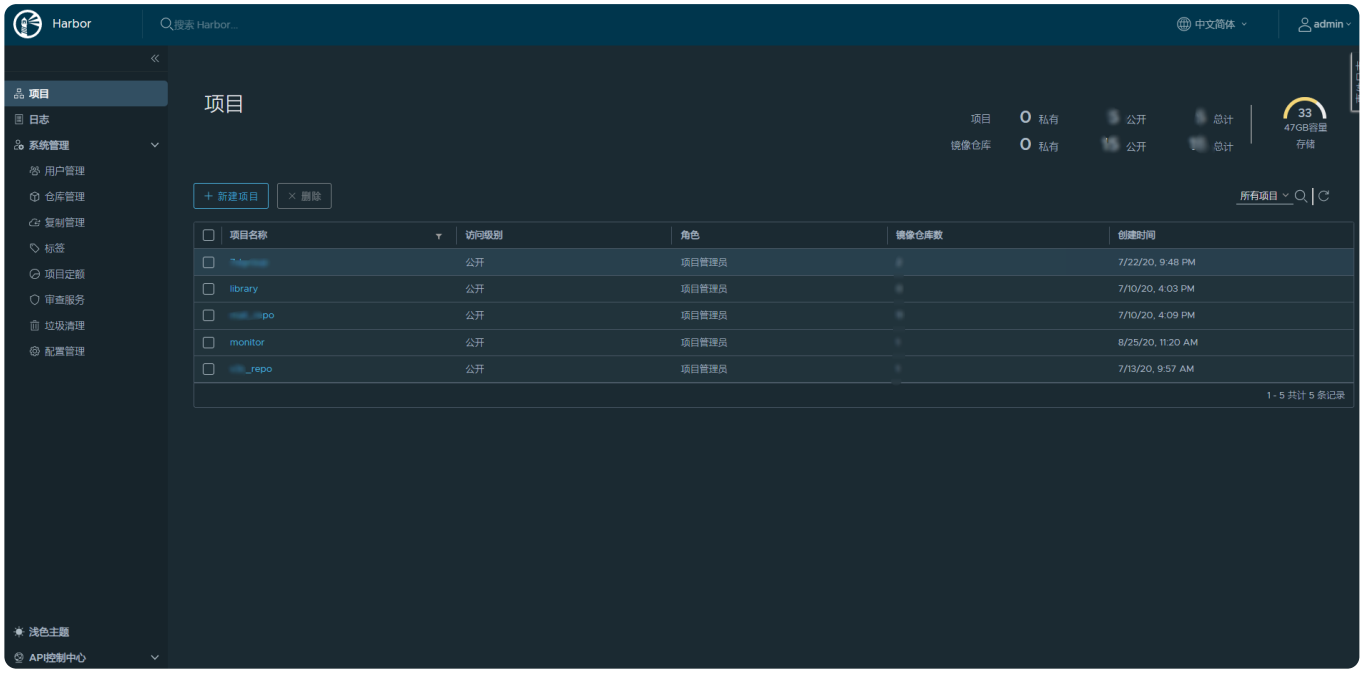
配置管理：


NACOS.
首页 文档 博客 社区 En nacos
编辑配置
* Data ID: mall-search.yaml
* Group: DEFAULT_GROUP
更多高级选项
描述:
Beta发布: 默认不要勾选。
配置格式: TEXT JSON XML YAML HTML Properties
配置内容:
server:
port: 8081
tomcat:
basedir: /var/tmp
background-processor-delay: 30
redirect-context-root: true
uri-encoding: UTF-8
accesslog:
enabled: true
buffered: true
directory: ./log
file-date-format: .yyyy-MM-dd
pattern: '%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-Agent}i" %D ms %F ms'
prefix: access_log
rename-on-rotate: false
request-attributes-enabled: false
发布
返回

系统保护：



容器仓库：



3. 资源监控

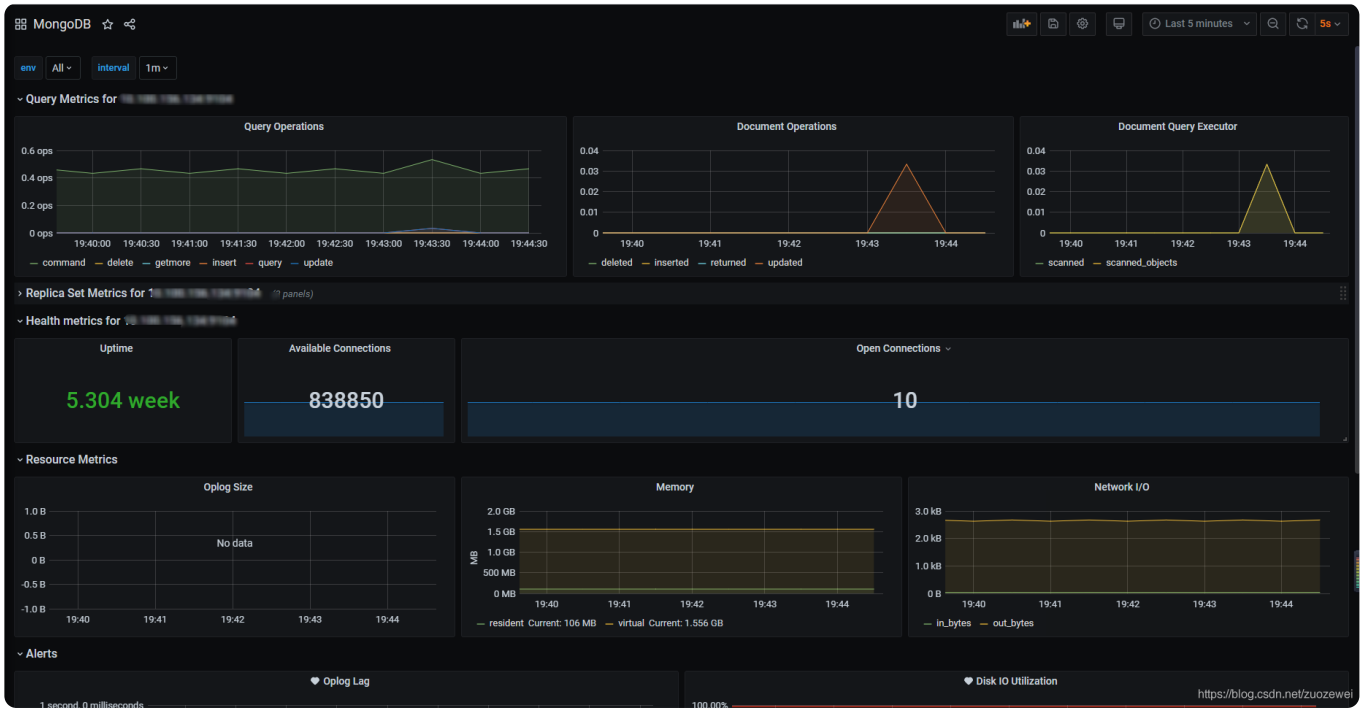
Kubernetes 集群资源监控：

Linux 资源监控：

MySQL 资源监控：

RabbitMQ 资源监控：

MongoDB 数据库资源监控：



Kubernetes etcd 资源监控：

Kubernetes API Server 资源监控：

Kubernetes 服务拓扑：

总结

这节课的内容包括了云环境的说明、技术组件的具体搭建过程、示例系统的搭建过程以及运行效果。经过上面所有的步骤，我们就把整个课程涉及的所有技术组件、示例系统完全搭建起来了。

而我之所以选择这样的技术栈，主要有三方面的考虑：

1. **核心优势**

任务调度：为集群系统中的任务提供调度服务，自动将服务按资源需求分配到资源限制的计算节点；

资源隔离：为产品提供管控与服务节点隔离能力，保证研发应用和管控服务不产生相互的影响；

高可用能力：自动监控服务运行，根据运行情况对失效的服务进行自动重启恢复；

网络互联互通能力：提供统一的 IP 地址分配和网络互通能力；

统一编排管理能力：结合 Gitlab 和 k8s ，对输出的产品进行统一的编排管理；

公共产品组件可以为团队提供统一部署、验证、授权、调度和管控能力，为私有云服务提供基础性的支撑。

2. 核心设施平台（ IaaS 云）

提供计算、网络、存储等核心资源设备的虚拟化；

支持不同操作系统，包括主流的 Win 和 Linux 系统；

提供主要的三种服务：云主机、云网络、云硬盘；

提供可视化 Web UI；

提供 k8s 集群（容器云）规划、部署和运营；

支持多种计算、存储和网络方案。

3. 基础服务平台（ PaaS 云）

提供数据存储：支持常见 NFS 、Ceph RBD、Local Volume 等；

提供应用服务：支持自愈和自动伸缩、调度和发布、负载均衡等；

提供运维管理：支持日志监控、资源监控、消息告警等。

我们这个系统采用的技术栈，是当前技术市场中流行的主流技术栈，这样的环境具有很高的借鉴价值。而且，从我们要表达的 RESAR 全链路压测和逻辑来说，也说明 RESAR 全链路压测理念是足以支撑当前的技术栈的。

有了主流技术栈的技术能力，再有我们的性能分析逻辑思维的落地，不能说天下无双，也是够你横霸一方，独挡一面的了。

这节课的内容非常多，希望你能够好好消化吸收。我在这里只是给你提供了我们项目的选择和理由，我们项目的选择可能不是最适合你的，如果你有其他想法，也欢迎在留言区和我交流讨论。另外在文稿的最后，我还给你汇总了你可能想要参考的文章链接，希望能让你有所收获。

课后题

学完这节课，请你思考两个问题：

1. 你有没有搭建过云原生系统，谈谈你对云原生系统搭建的心得！
2. 引入 k8s 后，你觉得企业内的基础设施最大的变化是什么？

欢迎你在留言区与我交流讨论。当然了，你也可以把这节课分享给你身边的朋友，他们的一些想法或许会让你有更大的收获。我们下节课见！

参考链接汇总

1. Kubernetes 集群搭建：

   单 Master 集群：使用 kubeadm 安装单 master kubernetes 集群（脚本版）

   高可用方案：Kubernetes 高可用集群落地二三事

2. Kubernetes 的使用基础：

   Kubernetes 集群基本概念

   k8s 入门篇 -Kubernetes 的基本概念和术语

   k8s 命令篇 -Kubernetes 工作实用命令集结号

   Kubernetes 集群常用操作总结

3. Kubernetes 网络插件选型：Kubernetes 网络插件（CNI）超过 10Gbit/s 的基准测试结果
4. Kubernetes 存储插件：

   NFS 部署： Kubernetes 如何安装 NFS-Subdir-External-Provisioner 存储插件？

   Ceph 部署： Kubernetes 集群分布式存储插件 Rook Ceph 部署

5. Kubernetes 下的 MySQL 部署：如何在 Kubernetes 集群中搭建一个复杂的 MySQL 数据库
6. Elasticsearch 集群的部署：Kubernetes Helm3 部署 Elasticsearch & Kibana 7 集群
7. 镜像仓库 Harbor 的部署：Kubernetes 集群仓库 harbor Helm3 部署
8. Nacos 的部署：

   Docker 单机模式： Nacos Docker 快速开始

   Kubernetes：Kubernetes 集群部署 nacos 1.4（集群版）

9. Redis、RabbitMQ、MongoDB 单机部署的部署：

   Kubernetes 集群部署 Redis + redis_exporter (单点)

   Kubernetes 集群部署 RabbitMQ + exporter (单节点)

   Kubernetes 集群部署 MongoDB+exporter (单节点)

10. Logstash 的部署：整合 ELK 实现日志收集
11. Kubernetes 集群资源监控的部署：

    Kubernetes 集群监控 kube-prometheus 部署

    Kubernetes 集群监控 controller-manager & scheduler 组件

    Kubernetes 集群监控 ETCD 组件

12. 日志聚合部署的部署：Kubernetes 集群日志监控 EFK 安装
13. 依赖组件的部署：

    Kubernetes 集群监控 kube-prometheus 自动发现

    APM 链路跟踪的部署：Kubnernetes 集群部署 Zipkin+Kafka+ElasticSearch 实现链路追踪

14. 微服务项目介绍：《高楼的性能工程实战课》微服务电商项目技术全解析

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 24 | 压测平台：如何改造分布式调度平台？

下一篇 26 | 全局监控（上）：如何快速落地全局监控？

更多学习推荐

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。